Библиотека ограничений для спецификации индустриальных моделей данных

[1, 2]С.В. Морозов <serg@ispras.ru>
[1]Д.В. Ильин <denis.ilyin@ispras.ru>
[1, 3]В.А. Семенов <sem@ispras.ru>
[1, 2]О.А. Тарлапан <oleg@ispras.ru>
[1] Институт системного программирования РАН, 109004, Россия, г. Москва, ул. А. Солженицына, дом 25
[2] Московский государственный университет имени М.В. Ломоносова 119991 ГСП-1 Москва, Ленинские горы, МГУ имени М.В. Ломоносова, 2-й учебный корпус, факультет ВМК
[3]Московский физико-технический институт (государственный университет), 141700, Московская область, г. Долгопрудный, Институтский пер., 9

Аннотация. В статье проводится анализ спецификаций индустриально значимого семейства объектно-ориентированных моделей данных на языке EXPRESS, на основе которого выделяются паттерны ограничений целостности, используемые в них. Разрабатывается библиотека обобщенных функций на языке EXPRESS для представления каждого из паттернов, которая может применяться как при рефакторинге существующих моделей, так и при разработке новых. Использование паттернов ограничений в спецификациях моделей позволяет улучшить их наглядность, облегчить их дальнейшее сопровождение и развитие и, в целом, ускорить их разработку. Кроме того, появляется возможность их анализа автоматизированными средствами программной инженерии. Обсуждается возможность применения данной библиотеки для решения задачи верификации моделей. Работа поддержана РФФИ (грант 13-07-00390).

Ключевые слова: объектно-ориентированное моделирование, EXPRESS, STEP, IFC, CIS/2, паттерны ограничений, верификация моделей

DOI: 10.15514/ISPRAS-2015-27(4)-5

Для цитирования: Морозов С.В., Ильин Д.В., Семенов В.А., Тарлапан О.А. Библиотека ограничений для спецификации индустриальных моделей данных. Труды ИСП РАН, том 27, вып. 4, 2015 г., стр. 69-110. DOI: 10.15514/ISPRAS-2015-27(4)-5.

1. Введение

Модельно-ориентированный подход к разработке программного обеспечения (ПО) является одним из перспективных направлений программной инженерии. В его рамках предполагается активное использование моделей на всех этапах жизненного цикла сложных программных систем, включая их проектирование, разработку, сопровождение и развитие.

Важное место в данной области занимает методология MDA/MDD [1] и составляющие ее основу стандарты OMG: UML [2], MOF [3], XMI [4], CWM [5]. Их применение позволяет улучшить мобильность ПО за счет использования платформно-независимых моделей. В то же время, интерпретируемость моделей UML обеспечивает возможность их тестирования на ранних стадиях разработки, что позволяет повысить качество и надежность ПО.

При проектировании и производстве высокотехнологичной продукции в различных индустриальных областях с применением информационных технологий возникает проблема поддержки интероперабельности используемого ПО на различных этапах ее жизненного цикла, то есть способности одной программной системы открыто взаимодействовать с остальными без каких-либо ограничений. Подобная проблема решается путем разработки и стандартизации единой модели данных о продукции и единого интерфейса доступа к ним в рамках методологии STEP (STandard for Exchange of Product model data) [6], принятой в 1994 году в качестве международного стандарта по интероперабельности ПО. Таким образом, для поддержки интероперабельности ПО в различных индустриальных областях также используется модельно-ориентированный подход.

Согласно методологии STEP модель данных специфицируется на языке EXPRESS [7], а сами данные представляются в формате кодирования открытым текстом, известном также как STEP Physical File Format (SPFF) [8], в XML [9] или же в бинарном формате [10]. Форматы SPFF и STEP XML являются взаимозаменяемыми и преобразуются один в другой согласно установленным в стандарте правилам. STEP определяет также единый интерфейс доступа к данным [11]. Примерами таких моделей служат прикладные протоколы STEP для машиностроения [12], электроники [13], электротехники [14], автомобилестроения [15], судостроения [16–18], системотехники [19], производства мебели [20], а также стандартизованные независимо от STEP модели данных IFC для архитектуры и строительства [21], CIS/2 для строительства с использованием стальных конструкций [22], ISO 15926 для нефтегазодобывающей промышленности [23].

Прикладные протоколы STEP и независимые модели данных, разработанные на основе методологии STEP, представляют собой одну или несколько информационных схем на языке EXPRESS. Каждая схема включает описания объектных типов данных (ENTITY), каждый из которых характеризуется именем, уникальным в пределах схемы, набором атрибутов и ограничений

целостности. Данные представляют собой множество экземпляров объектных типов, специфицируемых в схемах EXPRESS. Ограничения целостности определяют условия соответствия этих данных семантике информационной модели. Часть ограничений, охватывающих несколько объектных типов, описываются в виде глобальных правил на уровне схемы. Остальные элементы, специфицируемые в схемах EXPRESS, носят вспомогательный характер. Так, определяемые пользователем типы данных, наряду с предопределенными типами языка EXPRESS, используются при описании атрибутов в объектных типах. Константы, процедуры и функции применяются в выражениях для вычисления значений производных атрибутов или проверки ограничений целостности данных. Детальное описание языка EXPRESS и его основных возможностей, а также классификацию его типов данных и ограничений можно найти в работах [24–26].

Обсуждаемые индустриальные модели данных являются масштабными и включают спецификации сотен или тысяч объектных типов данных и ограничений целостности. Они разрабатываются крупными консорциумами участников в течение продолжительного времени и требуют периодического пересмотра с целью их развития либо обнаружения и коррекции возможных ошибок. В связи с этим данные спецификации должны быть удобными как для чтения человеком, так и для их интерпретации и анализа автоматизированными средствами программной инженерии. С этой целью спецификации данных и ограничений целостности должны выражаться унифицированным образом.

Однако на практике даже семантически эквивалентные ограничения целостности, представляются, как правило, различными способами. Подобные неоднозначности вызываются богатством конструкций языка EXPRESS и могут проявляться даже в пределах одной модели, над которой трудились независимые группы разработчиков. Таким образом, систематизация ограничений, используемых в индустриальных моделях данных, и реализация их в виде единой библиотеки является актуальной задачей.

Целью проводимого исследования является анализ индустриально значимого семейства спецификаций, принятых в качестве международных или индустриальных стандартов на предмет возможности унификации представления в них ограничений целостности данных. В разделе 2 проводится аналитический обзор моделей данных, основанных на методологии STEP. В разделе 3 выделяются паттерны ограничений целостности данных, наиболее часто используемые в этих моделях. Для представления каждого из паттернов разрабатываются обобщенные функции на языке EXPRESS, организуемые в виде универсальной библиотеки. В разделе 4 приводятся рекомендации по использованию данной библиотеки. Детально обсуждаются возможности ее применения для решения задачи верификации моделей. В заключении подводятся итоги проведенного исследования.

2. Аналитический обзор моделей данных, основанных на методологии STEP

В данном разделе сначала рассматривается организация стандарта STEP, затем приводится пример архитектуры IFC как одной из независимых моделей, основанных на методологии STEP. Далее обсуждаются ограничения целостности данных, применяемые в моделях, специфицируемых на языке EXPRESS, и основные проблемы, связанные с их представлением и использованием.

2.1 Организация стандарта STEP

Стандарт STEP (ISO 10303), разрабатываемый и поддерживаемый техническим комитетом ISO TC 184, изначально предназначался для обеспечения интероперабельности САПР, применяемых в машиностроении. Однако впоследствии он распространился и на другие предметные области. Стандарт состоит из множества (нескольких сотен) томов, разрабатываемых и принимаемых по отдельности. Он имеет трехуровневую архитектуру, представленную на рис.1. Каждый из томов относится к одной из групп, представленных на данном рисунке.

<table>
<tr><td colspan="6">Application Protocols & Associated Abstract Test Suites</td></tr>
<tr><td colspan="6">Common Resources</td></tr>
<tr><td>Integrated generic resources</td><td>Integrated application resources</td><td>application-interpreted constructs</td><td>application modules</td><td>ISO 13584-20: logical model of expressions</td><td>ISO 15531-42: time model</td></tr>
<tr><td colspan="6">Methodology</td></tr>
<tr><td colspan="2">description methods</td><td colspan="2">implementation methods</td><td colspan="2">conformance testing methodology & framework</td></tr>
</table>

Рис. 1. Архитектура стандарта STEP (ISO 10303)

Методологическую часть стандарта STEP составляют три группы: методы описания, реализации и тестирования. К первой группе относятся том 1, описывающий структуру стандарта и определяющий основные понятия и термины, используемые в нем, том 11, содержащий основное руководство по языку моделирования данных EXPRESS, и том 14, определяющий язык EXPRESS-X [27] для преобразования данных, соответствующих одним схемам, специфицируемым на EXPRESS, в другие (аналогично языку QVT [28] для UML моделей или XSLT [29] для XML документов). Методы реализации (тома 21-29) описывают форматы данных STEP, стандартный интерфейс доступа к данным (SDAI) для организации STEP-совместимых баз данных и методы его связывания с языками программирования C, C++, Java, а также способы преобразования моделей на EXPRESS в UML/XMI. Методология тестирования (тома 31-35) описывает процедуры проверки соответствия прикладных систем стандарту, в частности определяет основные требования по тестированию, организацию данного процесса, абстрактные методы тестирования реализаций прикладных протоколов STEP и SDAI.

Общие ресурсы STEP — это спецификации наиболее общих элементов моделей данных, которые могут совместно использоваться несколькими прикладными протоколами. Фактически, они являются основными строительными блоками стандарта и существенно облегчают разработку новых моделей данных за счет повторного использования общих определений. К данной группе относятся интегрированные основные ресурсы (тома 41-61), интегрированные прикладные ресурсы (тома 101-112), прикладные компоненты (тома 501-523), прикладные модули (тома с нумерацией выше 1000), а также спецификации логической модели выражений и модели времени, заимствованные из других стандартов ISO.

Высший уровень иерархии архитектуры STEP составляют прикладные протоколы (тома серии 200) — модели данных, свойственные конкретной предметной области. Каждый прикладной протокол по требованиям стандарта должен состоять как минимум на 85% из объектных типов, наследуемых от общих ресурсов, и сопровождаться набором тестов для проверки соответствия ПО данному протоколу (тома серии 300). Детально архитектура STEP представлена в [24, 30].

2.2 Организация стандарта IFC

Industry Foundation Classes (IFC) — это стандартизованный формат данных с открытой спецификацией, предложенный, разработанный и поддерживаемый International Alliance for Interoperability (IAI) с целью обеспечения интероперабельности разнородного программного обеспечения в области архитектуры и строительства (Architecture, Engineering and Construction, AEC). Альянс был создан в 1995 году американскими и европейскими фирмами, работающими в AEC индустрии, вместе с фирмами-

производителями ПО для данной отрасли. Отделения IAI существуют в США, Великобритании, Франции, Германии, Финляндии, Испании, Японии, Сингапуре, Корее и Австралии. В 2005 году IAI был преобразован в buildingSMART International [31].

Рассмотрим организацию текущей версии стандарта IFC 4, которая принята также в качестве международного стандарта ISO 16739 [32]. Спецификация IFC включает определение схемы данных на языке EXPRESS, а также нормативно-справочной информации, в которую входят определения качественных (property) и количественных (quantity) характеристик объектов. Архитектура IFC представлена на рис. 2. Схема данных условно подразделяется на несколько подсхем, каждая из которых принадлежит одному из четырех концептуальных уровней:

1. Ресурсы (Resource) — включает множество подсхем, содержащих определения структур данных для общих ресурсов, совместно используемых на более высоких уровнях;
2. Ядро (Core) — определяет основную структуру иерархии, фундаментальные взаимоотношения и наиболее общие понятия объектной модели IFC как абстрактные типы данных, от которых прямо или косвенно наследуются объектные типы, объявленные на более высоких концептуальных уровнях;
3. Интероперабельность (Interoperability) — включает подсхемы, содержащие определения типов данных, которые используются в архитектурно-строительных процессах, продуктах и ресурсах на междисциплинарном уровне; объектные типы, объявленные в них, являются специализациями типов ядра IFC, но могут быть конкретизированы на уровне прикладных дисциплин либо использоваться в ассоциативных связях с объектными типами высшего концептуального уровня;
4. Сфера деятельности (Domain) — включает автономные подсхемы, содержащие определения типов данных, которые являются специфичными для конкретных прикладных дисциплин и применяются при обмене совместно используемой информацией между приложениями, относящимися к одной и той же области знания; объектные типы данных на этом уровне являются окончательными специализациями и не могут использоваться на прочих более низких уровнях.

Все объектные типы данных уровня ядра и выше прямо или косвенно наследуются от корневого объектного типа иерархии `IfcRoot`. В данном типе появляется определение глобального идентификатора (GUID) и, как следствие, понятие объектной идентичности (как совпадение значений GUID). В объектных типах общих ресурсов GUID не определен, и их экземпляры не могут существовать независимо от экземпляров высокоуровневых объектов. Таким образом, они обязаны прямо или косвенно

участвовать в ассоциациях с объектными типами, унаследованными от классов ядра IFC. Заметим, что часть схем ресурсов IFC использует спецификации основных ресурсов STEP для представления фундаментальной информации о продукте [33] (включая сведения о персонах и организациях, измерениях величин), геометрии и топологии [34], формы [35], материалов [36], визуальной информации [37]. В IFC эти спецификации были упрощены с учетом специфики прикладной области архитектуры и строительства.

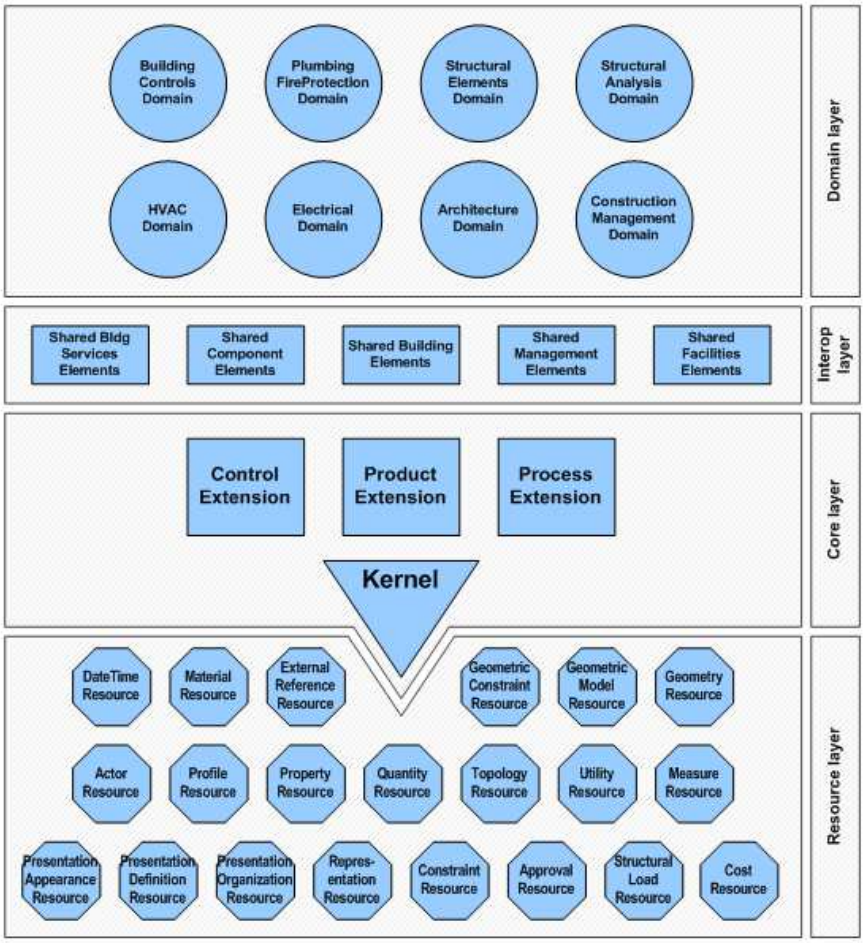

Рис. 2. Архитектура стандарта IFC 4

Заметим, что большинство механизмов, принятых в модели IFC, свойственно также известным онтологическим подходам [38]. Во-первых, это отдельные определения для объектных типов (наследуемых от абстрактного типа ядра

`IfcTypeObject`) и экземпляров или индивидов (наследуемых от абстрактного типа ядра `IfcObject`). Такой механизм позволяет легко расширить семантику модели путем создания новых экземпляров объектных типов и их ассоциации с соответствующими индивидами. Он широко применяется для уточнения типов различных управляющих элементов или же специализированного оборудования.

Во-вторых, в IFC используется объектификация взаимоотношений, которая позволяет сохранять их специфичные характеристики отдельно от типов взаимосвязанных объектов и расширить семантику языка EXPRESS путем определения иерархии типов взаимоотношений, наследуемых от абстрактного типа ядра `IfcRelationship`. Данная иерархия включает не только ассоциативные связи между объектами одного концептуального уровня, но и агрегации, композиции, группировки, определения и прочие зависимости, отсутствующие в языке EXPRESS.

В-третьих, каждому объекту IFC (`IfcObject`) и объектному типу (`IfcTypeObject`) могут быть назначены наборы качественных и количественных характеристик (`IfcPropertySetDefinition`), определяющие дополнительные статические свойства объектов и их типов наряду со стандартными атрибутами. Часть наборов характеристик для конкретных объектов и типов предопределена в стандарте. Любое из программных приложений, работающих с моделью IFC, может определять собственные динамически расширяемые наборы. В последнем случае стандарт предоставляет лишь метамодель для их описания: шаблон `IfcPropertyTemplateDefinition`, определяющий имена и описания характеристик, их типы данных, синтаксис для представления их значений, включая единицы измерения, и правила их назначения определенным объектам или типам. Приложение может предоставлять такой шаблон, но это является необязательным требованием стандарта.

2.3 Ограничения целостности данных, используемые в индустриальных моделях

Ограничения целостности данных, используемые в индустриальных моделях, специфицируемых на языке EXPRESS, можно условно подразделить на следующие группы:

- ограничения кардинальности (длин строковых и двоичных последовательностей, размеров коллекций, кратности прямых и инверсных связей);
- ограничения уникальности элементов в коллекциях или значений атрибутов объектов в наборе данных;
- требования наличия определенного значения для обязательных атрибутов объектов или элементов массивов;

- ограничения области определения значений, представляемые произвольными логическими выражениями или функциями.

Первые три группы ограничений представляются тривиальными синтаксическими конструкциями. В связи с этим, наибольший интерес для проводимого исследования составляет последняя группа ограничений, к которой относятся локальные правила WHERE, определяемые в простых и объектных типах данных и налагаемые на их индивидуальные экземпляры, а также глобальные правила RULE, определяемые на уровне схемы данных и налагаемые на экстенты одного или нескольких объектных типов.

Для спецификации локальных и глобальных правил могут использоваться разнообразные конструкции: определяемые пользователем функции, возвращающие результат логического типа, а также логические выражения. Несмотря на то, что язык EXPRESS является декларативным, в теле функций могут применяться операторы, свойственные императивным языкам, включая присвоения, условные переходы и циклы. Выражения языка EXPRESS также отличаются богатством синтаксических элементов (см. рис. 3). Используемые символы для записи операций в большинстве случаев соответствуют применяемым в других декларативных и императивных языках. Отдельно укажем только специфические для EXPRESS операции:

- *:=:* и *:<>:* — идентичность и неидентичность экземпляров объектного типа;
- || — конструирование экземпляра сложного объектного типа (конкатенирует конструкторы простых типов);
- \ — групповая ссылка (возвращает ссылку на значение подтипа внутри экземпляра сложного объекта, включающую значения всех атрибутов, определенных в данном подтипе);
- <= и >= для типов коллекций — подмножество и надмножество (проверка, является ли первая коллекция, соответственно, подмножеством или надмножеством второй);
- +, *, - для типов коллекций — теоретико-множественные операции.

Таким образом, разработчик спецификаций на языке EXPRESS может использовать богатый репертуар сложных семантических конструкций для определения локальных и глобальных правил. Как результат, одно и то же ограничение целостности данных может быть выражено несколькими альтернативными способами. Это может быть продемонстрировано на конкретном примере.

```
expression := simple_expression [ rel_op simple_expression ]
```



```
rel_op := '<' | '>' | '<=' | '>=' | '<>' | '=' | ':<>:' |
':=:' | IN | LIKE
simple_expression := term { add_like_op term }
term := factor { multiplication_like_op factor }
factor := simple_factor [ '**' simple_factor ]
simple_factor := aggregate_initializer | entity_constructor |
                    enumeration_reference | interval |
query_expression |
                    ( [ unary_op ] ( '(' expression ')' |
primary ) )
unary_op := '+' | '-' | NOT
primary := literal | (qualifiable_factor { qualifier } )
multiplication_like_op := '*' | '/' | DIV | MOD | AND | '||'
add_like_op = '+' | '-' | OR | XOR
interval := '{' simple_expression interval_op
simple_expression interval_op
                    simple_expression '}'
interval_op := '<' | '<='
query_expression := QUERY '(' variable_id '<*'
aggregate_source '|'
                          logical_expression ')'
aggregate_source := simple_expression
aggregate_initializer := '[' [ element { ',' element } ] ']'
element := expression [ ':' numeric_expression ]
enumeration_reference := [ type_id '.' ] enumeration_id
entity_constructor := entity_id '(' [ expression { ','
expression } ] ')'
qualifiable_factor := attribute_id | built_in_constant |
constant_id |
                    function_call | parameter_id | variable_id
| entity_id
qualifier := attribute_qualifier | group_qualifier |
index_qualifier
```

```
attribute_qualifier := '.' atttibute_id
group_qualifier := '\' entity_id
index_qualifier := '[' numeric_expression [ ':'
numeric_expression ] ']'
function_call := ( built_in_function | function_id ) [
actual_parameter_list ]
actual_parameter_list := '(' expression { ',' expression } ')'
```

Рис. 3. Синтаксис выражений языка EXPRESS

В модели IFC наиболее часто используемым ограничением является уточнение типов объектов, участвующих в ассоциациях. Рассмотрим спецификации двух объектных типов данных, принадлежащих подсхеме Geometric Model Resources: IfcBooleanClippingResult, представляющий твердотельный объект, получаемый в результате отсечения булевскими операциями и IfcPolygonalBoundedHalfSpace, представляющий полупространство, ограниченное полигоном (см. рис. 4). В первом типе ограничиваются типы данных операндов для булевской операции. Так, первый операнд может быть твердым телом, получаемым либо путем перемещения произвольного плоского сечения в пространстве, либо путем перемещения окружности вдоль трехмерной директрисы, либо в результате другого отсечения (см. правило FirstOperandType). Во втором типе устанавливается, что граница полупространства может быть либо ломаной линией, либо композитной кривой (см. правило BoundaryType). При этом первое ограничение выражается через дизъюнкцию операций принадлежности имен отдельных допустимых типов множеству названий типов данных, членом которых является объект. Второе — через функцию проверки мощности множества, получаемого в результате пересечения множеств имен допустимых типов и названий типов данных, членом которых является объект.

```
ENTITY IfcBooleanClippingResult SUBTYPE OF (IfcBooleanResult);
 WHERE
      FirstOperandType : ('IFC4.IFCSWEPTAREASOLID' IN
TYPEOF(FirstOperand)) OR
                         ('IFC4.IFCSWEPTDISCSOLID' IN
TYPEOF(FirstOperand)) OR
                         ('IFC4.IFCBOOLEANCLIPPINGRESULT' IN
TYPEOF(FirstOperand));
```



```
      SecondOperandType : ('IFC4.IFCHALFSPACESOLID' IN
TYPEOF(SecondOperand));
      OperatorType : Operator = DIFFERENCE;
END_ENTITY;

ENTITY IfcPolygonalBoundedHalfSpace SUBTYPE OF
(IfcHalfSpaceSolid);
      Position : IfcAxis2Placement3D;
      PolygonalBoundary : IfcBoundedCurve;
 WHERE
      BoundaryDim : PolygonalBoundary.Dim = 2;
      BoundaryType : SIZEOF(TYPEOF(PolygonalBoundary) * [
                        'IFC4.IFCPOLYLINE',
                        'IFC4.IFCCOMPOSITECURVE']
                      ) = 1;
END_ENTITY;
```

Рис. 4. Примеры спецификации локальных правил в модели IFC

Подобные неоднозначности представления эквивалентных ограничений целостности данных затрудняют их интерпретацию как человеком, так и автоматизированными средствами. В связи с этим, является важной систематизация ограничений и унификация их представления.

3. Паттерны ограничений в индустриальных моделях данных

Проведенный анализ моделей данных IFC, CIS/2, а также общих ресурсов STEP показал, что, несмотря на большое количество ограничений, используемых в этих моделях (несколько сотен в каждой), все они соответствуют небольшому числу типовых случаев (паттернов). В перечисленных моделях возможно выделить 17 паттернов, каждый из которых соответствует определенному семантическому ограничению, но может быть выражено разными синтаксическими формами. Тем не менее, паттерны обычно легко распознаются по наличию тех или иных конструкций языка EXPRESS в логическом выражении ограничения.

Ниже детально обсуждается каждый из выделенных паттернов ограничений по следующей схеме. Выделяются типовые конструкции языка EXPRESS, которые соответствуют данному паттерну. Проводится математическая

формализация и по возможности предлагается обобщенная функция на языке EXPRESS для выражения ограничений данного типа.

3.1 Set Pattern

Ограничения данного вида устанавливают принадлежность значения переменной некоторому конечному множеству: $v \in S$. На языке EXPRESS они записываются с помощью оператора принадлежности IN, применяемого к атрибутам перечисляемого или строкового типа либо в виде дизъюнкции операций равенства атрибута допустимым литеральным значениям. Конечные множества строк наиболее часто используются в индустриальных моделях данных для задания названий предопределенных цветов, стилей оформления, семейств шрифтов, форматов представления данных и т.п. В обобщенном виде подобные ограничения можно представить следующей функцией:

```
FUNCTION BelongsToSet(v: GENERIC:t; s: AGGREGATE OF GENERIC:t)
: LOGICAL;
RETURN (v IN s);
END_FUNCTION;
```

Заметим, что к данному паттерну также можно отнести условия неравенства переменных перечисляемого типа некоторому литералу: $v \neq const$. Они легко преобразуются к виду $v \in S'$, где $S' \cup \{const\} = S, const \notin S'$. В качестве примера рассмотрим спецификацию объектного типа `IfcStructuralCurveReaction` для представления реакций (сил, смещений, отклонений и т.п.), распределенных вдоль кривой. Здесь правило `SuitablePredefinedType` запрещает синусоидальные и параболические типы распределений.

```
TYPE IfcStructuralCurveActivityTypeEnum = ENUMERATION OF
      (CONST, LINEAR, POLYGONAL, EQUIDISTANT, SINUS,
PARABOLA, DISCRETE,
            USERDEFINED, NOTDEFINED);
END_TYPE;

ENTITY IfcStructuralCurveReaction SUBTYPE OF
(IfcStructuralReaction);
      PredefinedType : IfcStructuralCurveActivityTypeEnum;
 WHERE
```



```
      HasObjectType : (PredefinedType <>
IfcStructuralCurveActivityTypeEnum.USERDEFINED) OR
EXISTS(SELF\IfcObject.ObjectType);
      SuitablePredefinedType : (PredefinedType <>
IfcStructuralCurveActivityTypeEnum.SINUS) AND (PredefinedType
<> IfcStructuralCurveActivityTypeEnum.PARABOLA);
END_ENTITY;
```

Данное правило может быть легко переформулировано с использованием предложенной обобщенной функции:

```
      SuitablePredefinedType :
BelongsToSet(SELF.PredefinedType,
                                    [
IfcStructuralCurveActivityTypeEnum.CONST,

      IfcStructuralCurveActivityTypeEnum.LINEAR,

      IfcStructuralCurveActivityTypeEnum.POLYGONAL,

      IfcStructuralCurveActivityTypeEnum.EQUIDISTANT,

      IfcStructuralCurveActivityTypeEnum.DISCRETE,

      IfcStructuralCurveActivityTypeEnum.USERDEFINED,

      IfcStructuralCurveActivityTypeEnum.NOTDEFINED ]);
```

3.2 Range Pattern

Данные условия ограничивают область определения значений числовых переменных (целых или вещественных) интервалом или полуинтервалом: замкнутым ($m \leq v \leq n$, или $v \leq n$, или $v \geq m$), открытым ($m < v < n$, или $v < n$, или $v > m$) или полуоткрытым ($m < v \leq n$ или $m \leq v < n$). На языке EXPRESS они выражаются с помощью операторов сравнения или оператора интервала. В обобщенном виде их можно представить с помощью следующей функции, в которой значение переменной и границы интервала имеют общий

числовой тип NUMBER, а для определения замкнутости границ дополнительно вводятся два формальных параметра булевского типа:

```
FUNCTION LiesInRange(v, m, n : NUMBER; closedM, closedN :
BOOLEAN) : LOGICAL;
LOCAL
	resultLow : LOGICAL := TRUE;
	resultHigh : LOGICAL := TRUE;
END_LOCAL;
IF (EXISTS(m)) THEN
	IF (closedM = TRUE) THEN
		resultLow := m <= v;
	ELSE
		resultLow := m < v;
	END_IF;
END_IF;
IF (EXISTS(n)) THEN
	IF (closedN = TRUE) THEN
		resultHigh := v <= n;
	ELSE
		resultHigh := v < n;
	END_IF;
END_IF;
RETURN (resultLow AND resultHigh);
END_FUNCTION;
```

Неравенства вида $v \neq const$, где v — числовая переменная, представляются в виде дизъюнкции двух открытых полуинтервалов: $v < const \vee v > const$. Следовательно, их можно выразить в виде комбинации двух паттернов интервала, связанных логической функцией OR: `LiesInRange(v, ?, const, FALSE, FALSE) OR LiesInRange(v, const, ?, FALSE, FALSE)`, где символ '?' соответствует неопределенному значению.

Если ограничения интервала определены для коллекции чисел, то их можно переформулировать в виде следующей обобщенной функции:

```
FUNCTION AggregateLiesInRange(v: AGGREGATE OF NUMBER; m, n :
NUMBER; closedM, closedN : BOOLEAN): LOGICAL;
```



```
RETURN (SIZEOF(QUERY(temp <* v | LiesInRange(temp, m, n,
closedM, closedN))) = SIZEOF(v));
END_FUNCTION;
```

3.3 Constantation Pattern

Ограничения данного вида применяются с целью уточнения значения атрибута константой: $v = const$. Такое уточнение обычно используется в производных объектных типах, когда специализация допускает единственное конкретное значение для унаследованного атрибута. Также уточнение константой может происходить и для атрибутов ассоциируемых объектов при установлении соответствующих связей. Последнее наиболее часто используется при определении геометрических типов данных, когда требуется установить одинаковую размерность для геометрических объектов, связанных с некоторыми другими. Подобные ограничения можно представить в виде следующей обобщенной функции:

```
FUNCTION EqualsToConstant(v: GENERIC:t; const: GENERIC:t) :
LOGICAL;
RETURN (v = const);
END_FUNCTION;
```

Заметим, что подобные случаи близки по смыслу к уточнению атрибута в специализации объектного типа данных. Однако при генерации объектов уточненные атрибуты пропускаются и в результирующем наборе данных при его представлении в формате SFF на месте их значений ставится символ ‘*’. При наличии же ограничения константации значение соответствующего атрибута генерируется и представляется требуемым литералом в результирующем наборе.

3.4 Multiplicity Pattern

Ограничения данного вида уточняют размер атрибутов-коллекций, устанавливая либо их минимально допустимый ($|c| \geq size$), либо точный размер ($|c| = size$) в специализациях объектных типов данных или при установлении ассоциативных связей между объектами геометрических типов (аналогично ограничениям константации). Паттерн распознается по использованию функции SIZEOF для определения размера коллекции. В обобщенном виде это можно выразить с помощью следующей функции, где формальный параметр булевского типа `exact` устанавливает, задается ли точный или минимальный размер:

```
FUNCTION IsSizePermissible(c: AGGREGATE OF GENERIC; size:
INTEGER, exact: BOOLEAN): LOGICAL;
```

```
IF (exact = TRUE) THEN
      RETURN (SIZEOF(c) = size);
ELSE
      RETURN (SIZEOF(c) >= size);
END_IF;
END_FUNCTION;
```

Заметим, что иногда в спецификациях вместо функции SIZEOF при определении размеров коллекций используется функция HIINDEX, которая эквивалентна SIZEOF для списков, множеств и мультимножеств. В тех случаях, когда допустимый размер определяется строгим неравенством ($|c| > size$), его следует преобразовать в нестрогое ($|c| \geq size + 1$) для возможности выражения с помощью вышеприведенной обобщенной функции.

Для строковых данных, минимально допустимая или точная длина которых в EXPRESS определяется с помощью функции LENGTH, обобщенная функция может быть представлена в следующем виде:

```
FUNCTION IsStringSizePermissible(v: STRING; size: INTEGER,
exact: BOOLEAN): LOGICAL;
IF (exact = TRUE) THEN
      RETURN (LENGTH(v) = size);
ELSE
      RETURN (LENGTH(v) >= size);
END_IF;
END_FUNCTION;
```

К этому же паттерну следует отнести ограничения длин двоичных последовательностей, определяемые в EXPRESS с помощью функции BLENGTH и устанавливающие требования их кратности восьми или целому числу байтов:

```
FUNCTION IsIntegerNumberOfBytes(v: BINARY) : LOGICAL;
RETURN (BLENGTH(v) MOD 8 = 0);
END_FUNCTION;
```

3.5 Existence Pattern

Ограничения данного вида либо устанавливают обязательность существования значения опционального атрибута, наследуемого от базового объектного типа ($\exists v$), либо контролируют существование значений для

нескольких атрибутов одного объектного типа, определяемое некоторой логической функцией. В первом случае логическое выражение на языке EXPRESS состоит из единственной функции EXISTS и его можно представить в следующем обобщенном виде:

```
FUNCTION IsExist(v: GENERIC) : LOGICAL;
RETURN (EXISTS(v));
END_FUNCTION;
```

Во втором случае функции EXISTS для проверки существования значений индивидуальных атрибутов связываются допустимыми в языке логическими операциями: AND, OR, XOR, NOT. Подобные ограничения можно рассматривать как комбинацию паттернов существования.

3.6 Existence on Determinator Pattern

Ограничения данного вида устанавливают требование существования значения для опционального атрибута в зависимости от состояния другого атрибута этого объектного типа данных, который называется определителем и имеет перечисляемый тип: $d = const \rightarrow \exists v$. Иногда определитель может также являться опциональным. Тогда спецификация ограничения включает дополнительную проверку существования его значения: $\exists d \wedge d = const \rightarrow \exists v$.

Заметим, что данные правила можно представить в виде комбинации паттернов существования и константации. Однако это один из наиболее часто встречаемых видов ограничений в индустриальных моделях данных, в связи с чем было решено выделить их в отдельный паттерн. Подобные правила используются, например, в вышеописанной схеме расширения семантики IFC для определяемых пользователем типов объектов. Рассмотрим тип данных `IfcBeam` для представления строительных балок. В нем атрибут `PredefinedType` является определителем, а строковый атрибут `ObjectType`, унаследованный от `IfcObject` и задающий имя определяемого пользователем типа балки — зависимым. Имя типа обязано существовать, если `PredefinedType` установлен как `USERDEFINED` (см. правило `CorrectPredefinedType`).

```
TYPE IfcBeamTypeEnum = ENUMERATION OF
      (BEAM, JOIST, HOLLOWCORE, LINTEL, SPANDREL, T_BEAM,
USERDEFINED,
            NOTDEFINED);
END_TYPE;
```

```
ENTITY IfcBeam SUPERTYPE OF (ONEOF (IfcBeamStandardCase))
SUBTYPE OF (IfcBuildingElement);
      PredefinedType : OPTIONAL IfcBeamTypeEnum;
 WHERE
      CorrectPredefinedType : NOT(EXISTS(PredefinedType)) OR
            (PredefinedType <> IfcBeamTypeEnum.USERDEFINED)
OR
            ((PredefinedType = IfcBeamTypeEnum.USERDEFINED)
AND EXISTS (SELF\IfcObject.ObjectType));
      CorrectTypeAssigned : (SIZEOF(IsTypedBy) = 0) OR
       ('IFC4.IFCBEAMTYPE' IN
TYPEOF(SELF\IfcObject.IsTypedBy[1].RelatingType));
END_ENTITY;
```

Подобные ограничения можно выразить следующей обобщенной функцией, где формальный параметр булевского типа `isDExists` устанавливает необходимость проверки существования значения определителя:

```
FUNCTION ExistsOnDeterminator(v: GENERIC:obj; d: GENERIC:t;
const: GENERIC:t; isDExist: BOOLEAN) : LOGICAL;
IF (isDExist = TRUE AND NOT(EXISTS(d))) THEN
      RETURN (TRUE);
END_IF;
IF (d <> const) THEN
      RETURN (TRUE);
END_IF;
RETURN (d = const AND EXISTS(v));
END_FUNCTION;
```

3.7 Type Refinement Pattern

Правила данного вида уточняют типы объектов, участвующих в ассоциациях. Если ассоциация определена на некотором обобщенном объектом типе, то с помощью данных правил можно ограничить количество специализаций, имеющих право участвовать в ней. Подобное уточнение обычно используется, если ассоциация унаследована от базового объектного типа. Формализовать данный тип ограничений можно следующим образом. Пусть функция $typeof(o)$ возвращает множество названий типов данных, членом

которых является объект o. То есть, это множество включает название собственного объектного типа и всех его обобщений вплоть до корня иерархии. Пусть множество T_A включает названия типов данных, допустимых для участия в ассоциации, а множество T_D — названия недопустимых типов. Тогда ограничения формулируются следующим образом: $|typeof(o) \cap T_A| = 1 \wedge |typeof(o) \cap T_D| = 0$. На языке EXPRESS это можно представить следующей обобщенной функцией, где дополнительный параметр `isAllowed` устанавливает, задается ли параметром t множество допустимых или недопустимых типов:

```
FUNCTION RefinesType(o: GENERIC; t: SET OF STRING; isAllowed:
BOOLEAN) : LOGICAL;
LOCAL
      allowedNumber : INTEGER;
END_LOCAL;
IF (isAllowed = TRUE) THEN
      allowedNumber := 1;
ELSE
      allowedNumber := 0;
END_IF;
RETURN (SIZEOF(TYPEOF(o) * t) = allowedNumber);
END_FUNCTION;
```

Для уточнения типов во множественных ассоциациях может использоваться следующая обобщенная функция:

```
FUNCTION RefinesTypeInCollection(c: AGGREGATE OF GENERIC; t:
SET OF STRING; isAllowed: BOOLEAN) : LOGICAL;
RETURN (SIZEOF(QUERY(temp <* c | RefinesType(temp, t,
isAllowed)) = SIZEOF(c));
END_FUNCTION;
```

3.8 Type Refinement on Determinator Pattern

Ограничения данного вида уточняют типы объектов, участвующих в ассоциациях, в зависимости от состояния атрибута-определителя. Их можно представить в виде комбинации паттернов константации и уточнения типа: $d = const \rightarrow |typeof(o) \cap T_A| = 1$. В связи с тем, что они также часто встречаются в индустриальных моделях данных, было решено выделить их в отдельный паттерн.

Как правило, подобные ограничения выражаются в виде логической функции, тело которой состоит из единственного оператора альтернативы CASE или же последовательности условных операторов, которые ставят в соответствие множества допустимых типов различным значениям атрибута-определителя. В анализируемых моделях данных ограничения подобного типа, как правило, применяются к коллекциям объектов. Тогда обобщенная функция на языке EXPRESS может быть представлена следующим образом:

```
FUNCTION RefinesTypeOnDeterminator(c: AGGREGATE OF
GENERIC:obj; d: GENERIC:dt; const: LIST OF GENERIC:dt; t: LIST
OF SET OF STRING) : LOGICAL;
REPEAT i := LOINDEX(const) TO HIINDEX(const);
      IF (d = const[i]) THEN
            IF (SIZEOF(t[i]) = 0) THEN
                  RETURN (TRUE);
            ELSE
                  RefinesTypeInCollection(c, t[i], TRUE);
            END_IF;
      END_IF;
END_REPEAT;
RETURN (?);
END_FUNCTION;
```

3.9 Subset Pattern

Ограничения данного вида устанавливают, является ли одна коллекция подмножеством другой: $v_1 \subseteq v_2$. На языке EXPRESS они формулируются либо с помощью операции <= над коллекциями, либо путем выполнения операции QUERY к элементам первой коллекции с условием проверки принадлежности каждого из ее элементов второй коллекции с помощью операции IN. Это можно выразить с помощью следующей обобщенной функции:

```
FUNCTION IsSubset(v1: AGGREGATE OF GENERIC:t; v2: AGGREGATE OF
GENERIC:t) : LOGICAL;
RETURN SIZEOF(QUERY(temp <* v1 | temp IN v2)) = SIZEOF(v1)));
END_FUNCTION;
```


3.10 Correlated Size Pattern

Ограничения данного вида устанавливают равенство размеров двух атрибутов-коллекций в объектном типе данных: $|c_1| = |c_2|$. На языке EXPRESS для определения размеров коллекций используются функции SIZEOF или HIINDEX, и в обобщенном виде данный паттерн можно представить с помощью следующей функции:

```
FUNCTION AreSizesEqual(c1: AGGREGATE OF GENERIC; c2: AGGREGATE
OF GENERIC): LOGICAL;
RETURN (SIZEOF(c1) = SIZEOF(c2));
END_FUNCTION;
```

К этому же паттерну следует отнести правила, устанавливающие равенство длин строк или двоичных последовательностей. Поскольку для определения этих длин в EXPRESS используются функции LENGTH и BLENGTH соответственно, для их представления в обобщенном виде вводятся индивидуальные функции:

```
FUNCTION AreStringSizesEqual(v1: STRING; v2: STRING) :
LOGICAL;
RETURN (LENGTH(v1) = LENGTH(v2));
END_FUNCTION;


FUNCTION AreBinarySizesEqual(v1: BINARY; v2: BINARY) :
LOGICAL;
RETURN (BLENGTH(v1) = BLENGTH(v2));
END_FUNCTION;
```

3.11 Correlated Type Pattern

Подобные ограничения определяются для селективных типов данных. Согласно этим правилам атрибуты должны быть установлены либо в одинаковые, либо в разные допустимые типы данных по списку выбора соответствующих селективных типов. Это можно формализовать следующим образом: $typeof(v_1) = typeof(v_2)$ или $typeof(v_1) \neq typeof(v_2)$, где функция $typeof(v)$ возвращает текущий тип данных для переменной селективного типа. На языке EXPRESS подобные ограничения можно представить с помощью следующей обобщенной функции, где дополнительный параметр `isEqual` устанавливает, должны ли совпадать типы данных:

```
FUNCTION AreTypesCorrelate(v1: GENERIC:t; v2: GENERIC:t;
isEqual:BOOLEAN) : LOGICAL;
IF (isEqual = TRUE) THEN
      RETURN (TYPEOF(v1) = TYPEOF(v2));
ELSE
      RETURN (TYPEOF(v1) <> TYPEOF(v2));
END_IF;
END_FUNCTION;
```

Аналогичные ограничения могут быть также определены для коллекций селективных элементов. Тогда обобщенная функция имеет следующий вид:

```
FUNCTION AreTypesCorrelateInCollection(c: AGGREGATE OF
GENERIC; isEqual:BOOLEAN) : LOGICAL;
IF (isEqual = TRUE) THEN
      RETURN (SIZEOF(QUERY(temp <* c | TYPEOF(temp) <>
TYPEOF(c[1]))) = 0);
ELSE
      RETURN (SIZEOF(QUERY(temp <* c | TYPEOF(temp) <>
TYPEOF(c[1]))) = SIZEOF(c));
END_IF;
END_FUNCTION;
```

3.12 Alternative Reference Pattern

Данные ограничения запрещают связь с одним и тем же экземпляром из двух однотипных ассоциаций в некотором объектном типе или же рекурсивную связь (ассоциацию объекта с самим собой). Они распознаются по наличию операций идентичности или неидентичности в логическом выражении. Требования единственности роли ассоциируемых объектов используются в наследуемых от `IfcRelationship` типах данных. Рекурсивные связи запрещаются в типах данных сложных качественных и количественных характеристик, которые конструируются как множества других характеристик. Если обе ассоциации являются одиночными, то ограничение представляется следующей обобщенной функцией на языке EXPRESS:

```
FUNCTION AreDifferentInstances(r1: GENERIC:t; r2: GENERIC:t) :
LOGICAL;
RETURN (r1 :<>: r2);
```



```
END_FUNCTION;
```

Для проверки отсутствия экземпляра во множественной ассоциации можно использовать следующую обобщенную функцию:

```
FUNCTION  IsInstanceMissingInMultipleAssociation(r:  AGGREGATE
OF GENERIC:t; o: GENERIC:t) : LOGICAL;
RETURN (SIZEOF(QUERY(temp <* r | o :=: temp)) = 0);
END_FUNCTION;
```

3.13 Unique Name Pattern

Данные ограничения устанавливают требования уникальности имен объектов, участвующих во множественной ассоциации с некоторым другим объектом: $\forall o_i \in O, o_j \in O \rightarrow o_i.name \neq o_j.name, i \neq j$, где O — множество ассоциируемых объектов. В IFC 4 такие требования распространяются, например, на имена качественных и количественных характеристик в пределах одного набора или сложной характеристики. На языке EXPRESS подобные правила можно представить в виде следующей обобщенной функции:

```
FUNCTION AreNamesUnique(O: AGGREGATE OF GENERIC) : LOGICAL;
LOCAL
      Names : SET OF STRING := [];
END_LOCAL;
REPEAT i := LOINDEX(O) TO HIINDEX(O);
      Names := Names + O[i].Name;
END_REPEAT;
RETURN (SIZEOF(Names) = SIZEOF(O));
END_FUNCTION;
```

3.14 Equivalence Pattern

Ограничения данного вида устанавливают равенство значений двух атрибутов в одном объектном типе или же атрибутов в объектах, связанных ассоциацией друг с другом или третьим объектом: $v_1 = v_2$. Обобщенная функция на языке EXPRESS для выражения подобных правил выглядит следующим образом:

```
FUNCTION AreAttributesEqual(v1: GENERIC:t; v2: GENERIC:t) :
LOGICAL;
RETURN (v1 = v2);
```

```
END_FUNCTION;
```

Еще одной разновидностью ограничений, которые можно отнести к данному паттерну, являются требования равенства какого-либо атрибута в объектах, участвующих во множественной ассоциации. Например, в IFC 4 эти правила устанавливают равенство размерностей ассоциируемых геометрических объектов или совпадение типов двухмерных сечений, что можно представить следующими обобщенными функциями:

```
FUNCTION AreDimensionsEqual(c: AGGREGATE OF GENERIC) :
LOGICAL;
RETURN (SIZEOF(QUERY(temp <* c | temp.Dim <> c[1].Dim)) = 0);
END_FUNCTION;

FUNCTION AreProfileTypesEqual(c: AGGREGATE OF GENERIC) :
LOGICAL;
RETURN (SIZEOF(QUERY(temp <* c | temp.ProfileType <>
c[1].ProfileType)) = 0);
END_FUNCTION;
```

3.15 Sequence Patterns

Ограничения данного вида устанавливают принадлежность значения атрибута некоторой аналитически определяемой последовательности. Можно выделить несколько типовых случаев в зависимости от специфики заданной последовательности.

3.15.1 Sequence Pattern: Loop

В данном случае устанавливается наличие циклической зависимости между элементами упорядоченной коллекции C, при которой ее первый элемент совпадает с последним: $c_1 = c_n,\ c_1, c_n \in C, |C| = n$. Для множественных ассоциаций требуется, чтобы первый и последний элемент указывали на один и тот же экземпляр объекта, что задается с помощью оператора идентичности :=: языка EXPRESS. Характерным примером использования данного вида ограничений являются объектные типы замкнутых ломаных или кривых. Ограничение представляется следующей обобщенной функцией:

```
FUNCTION IsLoop(c: AGGREGATE OF GENERIC) : LOGICAL;
RETURN c[1]:=:c[SIZEOF(c)];
END_FUNCTION;
```


3.15.2 Sequence Pattern: Regular Identity

Ограничение устанавливает условие идентичности соседних элементов коллекции, повторяющееся с некоторым шагом. К данному типу можно отнести, например, ограничения непрерывности топологических объектов типа путей или циклов, то есть требование совпадения вершин, соответствующих концу предыдущего и началу следующего ребра в этих объектах. В обобщенном виде этот тип ограничений возможно представить с помощью следующей функции:

```
FUNCTION IsRegularIdentity(c: AGGREGATE OF GENERIC, step:
INTEGER) : LOGICAL;
LOCAL
      N : INTEGER := 0;
      P : LOGICAL := TRUE;
END_LOCAL;
N := SIZEOF(c)-step;
REPEAT i := step TO N BY step;
      P := P AND (c[i] :=: c[i+1]);
END_REPEAT;
RETURN (P);
END_FUNCTION;
```

3.15.3 Sequence Pattern: Distinction

Данные правила устанавливают требование неравенства хотя бы одного из элементов коллекции заданному литеральному значению: $\exists c \in C: c \neq const$. Их можно выразить с помощью следующей обобщенной функции:

```
FUNCTION HasDistinctionFromGiven(c: AGGREGATE OF GENERIC:t,
const: GENERIC:t) : LOGICAL;
RETURN (SIZEOF(QUERY(v <* c | v <> const)) > 0);
END_FUNCTION;
```

3.16 Dependency Patterns

Данные ограничения выражают зависимость между значениями нескольких атрибутов некоторого объектного типа в виде произвольных неравенств: $f(x_1, \dots, x_n) \geq 0$. Они могут разрешаться на основе предварительно определенных правил вида: $x_1 = \tilde{f}_1(x_2, \dots, x_n) + C_1, \dots, x_n = \tilde{f}_n(x_1, \dots, x_{n-1}) + C_n$, где $C_i, i = 1, \dots, n$ — некоторые константы. Эти правила могут задаваться

вручную на основе расширения синтаксиса языка EXPRESS либо генерироваться автоматически по сигнатуре ограничений с помощью специальных методов.

В качестве примера рассмотрим объектный тип IfcCircleHollowProfileDef для представления круглых пустотелых сечений. В нем установлено ограничение WR1 на соотношение радиуса и толщины стенки отверстия. Тогда решающие правила могут быть заданы в виде синтаксической конструкции RULES FOR, расширяющий синтаксис языка EXPRESS. Зафиксировав значение одного из атрибутов, можно сгенерировать значение для другого с помощью данных правил. В приведенном ниже примере правила устанавливают, что толщина стенки отверстия составляет 1% от его радиуса.

```
ENTITY IfcCircleProfileDef SUPERTYPE OF (ONEOF
(IfcCircleHollowProfileDef))
 SUBTYPE OF (IfcParameterizedProfileDef);
       Radius : IfcPositiveLengthMeasure;
END_ENTITY;

ENTITY IfcCircleHollowProfileDef SUBTYPE OF
(IfcCircleProfileDef);
       WallThickness : IfcPositiveLengthMeasure;
 WHERE
       WR1 : WallThickness < SELF\IfcCircleProfileDef.Radius;
 RULES FOR WR1 :
       WallThickness = SELF\IfcCircleProfileDef.Radius * 0.01;
       SELF\IfcCircleProfileDef.Radius = WallThickness / 0.01;
 END_RULES;
END_ENTITY;
```

3.17 Complex Patterns

Подобные правила устанавливают ограничения целостности данных, задаваемые в виде сложных логических функций, сформулированных на языке EXPRESS. Обобщенное представление для данного семейства паттернов отсутствует, поскольку каждая такая функция является индивидуальной. В качестве примера рассмотрим объектный тип IfcBSplineCurveWithKnots для представления B-сплайн кривых с узлами, где правило корректной параметризации B-сплайна

ConsistentBSpline определяется функцией IfcConstraintsParamBSpline.

```
ENTITY IfcBSplineCurveWithKnots SUPERTYPE OF (ONEOF
       (IfcRationalBSplineCurveWithKnots)) SUBTYPE OF
(IfcBSplineCurve);
       KnotMultiplicities : LIST [2:?] OF INTEGER;
       Knots : LIST [2:?] OF IfcParameterValue;
       KnotSpec : IfcKnotType;
DERIVE
       UpperIndexOnKnots : INTEGER := SIZEOF(Knots);
 WHERE
       ConsistentBSpline : IfcConstraintsParamBSpline(Degree,
UpperIndexOnKnots,
                     UpperIndexOnControlPoints,
KnotMultiplicities, Knots);
       CorrespondingKnotLists : SIZEOF(KnotMultiplicities) =
UpperIndexOnKnots;
END_ENTITY;

FUNCTION IfcConstraintsParamBSpline( Degree, UpKnots, UpCp :
INTEGER;
       KnotMult : LIST OF INTEGER; Knots : LIST OF
IfcParameterValue ) : BOOLEAN;
LOCAL
       Result : BOOLEAN := TRUE;
       K, Sum : INTEGER;
END_LOCAL;
       Sum := KnotMult[1];
       REPEAT i := 2 TO UpKnots;
              Sum := Sum + KnotMult[i];
       END_REPEAT;
       IF (Degree < 1) OR (UpKnots < 2) OR (UpCp < Degree) OR
```

```
                        (Sum <> (Degree + UpCp + 2))
    THEN
            Result := FALSE;
            RETURN(Result);
    END_IF;
    K := KnotMult[1];
    IF (K < 1) OR (K > Degree + 1) THEN
            Result := FALSE;
            RETURN(Result);
    END_IF;
    REPEAT i := 2 TO UpKnots;
            IF (KnotMult[i] < 1) OR (Knots[i] <= Knots[i-1])
    THEN
                    Result := FALSE;
                    RETURN(Result);
            END_IF;
            K := KnotMult[i];
            IF (i < UpKnots) AND (K > Degree) THEN
                    Result := FALSE;
                    RETURN(Result);
            END_IF;
            IF (i = UpKnots) AND (K > Degree + 1) THEN
                    Result := FALSE;
                    RETURN(Result);
            END_IF;
    END_REPEAT;
    RETURN(result);
END_FUNCTION;
```

4. Использование библиотеки ограничений при решении задачи верификации моделей данных

Разработанная библиотека на языке EXPRESS, в состав которой входят перечисленные выше обобщенные функции для представления паттернов ограничений, может использоваться как при рефакторинге существующих

моделей, так и при разработке новых. Использование паттернов ограничений в спецификациях новых моделей позволяет улучшить их наглядность, облегчить их дальнейшее сопровождение и развитие и, в целом, ускорить их разработку. Кроме того, появляется возможность их анализа автоматизированными средствами программной инженерии.

Табл. 1. Методы разрешения паттернов ограничений целостности данных

Паттерн	Метод разрешения
Set	генерация значения, принадлежащего множеству допустимых
Range	генерация значения, лежащего в заданном интервале или полуинтервале
Constantation	генерация константного значения
Multiplicity	установка размера коллекции в соответствии с уточненным интервалом
Existence	генерация значений соответствующих атрибутов в обязательном порядке
Existence on Determinator	генерация значения определителя в первую очередь, затем в зависимости от него генерация значения для зависимого атрибута в обязательном или необязательном порядке
Type Refinement	выбор кандидатов ассоциируемых объектов из ограниченного соответствующим образом множества
Type Refinement on Determinator	генерация значения определителя в первую очередь, затем выбор кандидатов ассоциируемых объектов из ограниченного соответствующим образом множества
Subset	генерация значений элементов второй коллекции, затем заполнение первой коллекции некоторой случайной выборкой элементов из второй
Correlated Size	установка одинакового допустимого размера для обеих коллекций
Correlated	установка одного и того же типа данных из списка выбора

Type	селективного типа
Alternative Reference	установка одной из ассоциаций, затем выбор кандидатов для другой из ограниченного соответствующим образом множества
Unique Name	выбор кандидатов с уникальным именем либо генерация уникального имени при установлении ассоциативной связи
Equivalence	присвоение соответствующим атрибутам одинаковых значений
Sequence	генерация значений элементов коллекции в соответствии с заданной последовательностью
Dependency	поиск значения на основе интервальной арифметики и локального распространения
Complex	специальные методы генерации значений

Обсудим возможности применения этой библиотеки при решении задачи верификации моделей данных. В работе [39] был предложен комбинированный метод, применимый к анализу масштабных моделей данных. Он сначала разрешает сильно связанные ограничения кардинальности, уникальности и определяет необходимый размер семантически корректной коллекции объектов, а затем, изолированно обрабатывая индивидуальные ограничения, позволяет установить корректные значения атрибутов объектов и ассоциативные связи между ними, удовлетворяющие каждому из установленных ограничений.

На этапе генерации значений атрибутов и установления ассоциативных связей задача верификации моделей редуцируется к задаче удовлетворения ограничений (Constraint Satisfaction Problem, CSP) [40]. Методы ее решения предполагают наличие разрешающих правил для ограничений модели. Ручное создание подобных правил не представляется возможным вследствие масштабности индустриальных моделей данных, а их автоматизированное построение в общем случае является затруднительным, поскольку ограничения целостности данных, в особенности ограничения области определения значений, в языке EXPRESS могут представляться разнообразными синтаксическим конструкциями, включая определяемые пользователем функции и произвольные логические выражения. Таким образом, предлагается задействовать разработанную библиотеку ограничений и использовать предопределенные разрешающие правила для паттернов. Для существующих моделей данных следует предварительно провести

рефакторинг определенных в них ограничений с целью унификации их представления.

Для каждого из выделенных паттернов ограничений были разработаны методы разрешения. Примечательно, что для большинства из паттернов (15 из 17) эти методы оказались тривиальными и связанными с ограничением области определения при генерировании значений зависимых атрибутов (см. табл. 1).

Рассмотрим в качестве примера псевдокод функции для установки значений атрибутов в объектах типа `IfcPolygonalBoundedHalfSpace`, спецификация которого представлена на рис. 4:

```
function setIfcPolygonalBoundedHalfSpaceAttributes( o :
        IfcPolygonalBoundedHalfSpace )
begin_function
        if( not exists( o.Position ) )
        then
                IfcAxis2Placement3D pos :=
                        generate( "IFC4.IFCAXIS2PLACEMENT3D" );
                setIfcAxis2Placement3DAttributes( pos );
                o.Position := pos;
        end_if;
        if( not exists( o.PolygonalBoundary ) )
        then
                IfcBoundedCurve bound;
                (* select the type according to Where Rule
"BoundaryType" *)
                int typeN = rnd( 0, 1 );
                if( typeN = 0 )
                then
                        bound := generate( "IFC4.IFCPOLYLINE" );
                        (* set the value according to Where Rule
"BoundaryDim" *)
                        bound.Dim := 2;
                        setIfcPolylineAttributes( bound );
                else
```

```
                    bound := generate(
"IFC4.IFCCOMPOSITECURVE" );
                    (* set the value according to Where Rule
"BoundaryDim" *)
                    bound.Dim := 2;
                    setIfcCompositeCurveAttributes( bound );
            end_if;
            o.PolygonalBoundary := bound;
      end_if;
end_function;
```

Для разрешения ограничений, принадлежащих семейству паттернов Dependency, возможно применить методы локального распространения [41] при условии, что зависимости между переменными являются ациклическими. Способы конструирования и примеры разрешающих правил, необходимых для данных методов, были приведены в разделе 3.16.

Наиболее сложно разрешаются семейства паттернов ограничений Complex, представляемые произвольными логическими функциями. Однако в рассмотренных индустриальных моделях данных они не являются широко распространенными и встречаются в определениях геометрических объектов, которые в значительной степени стандартизированы. Несложно определить специальные методы генерации корректного геометрического представления, которые будут применимы к любой из моделей. Способы разрешения сложных ограничений в классах модели IFC 4 приведены в табл. 2.

Табл. 2. Способы разрешения сложных геометрических ограничений в модели IFC 4

Класс	Ограничение	Способ разрешения
IfcAxis2Placement3D (локальные системы координат)	AxisToRefDirPosition (оси не должны быть коллинеарными)	Создание взаимно перпендикулярных векторов ненулевой длины
IfcBSplineCurveWithKnots IfcBSplineSurfaceWithKnots (B-сплайн кривые и поверхности)	ConsistentBSpline UDirectionConstraints VDirectionConstraints (проверяется	Генерация B-сплайнов с помощью известных

	корректность параметризации B-сплайнов)	алгоритмов [42]
IfcExtrudedAreaSolid (3D объект, получаемый путем перемещения сечения в заданном направлении на фиксированное расстояние)	ValidExtrusionDirection (заданное направление не должно быть перпендикулярно оси Z в локальной системе координат)	Выбор направления перемещения сечения вдоль локальной оси Z
IfcExtrudedAreaSolidTapered (расширяет IfcExtrudedAreaSolid указанием концевой поверхности) IfcRevolvedAreaSolidTapered (3D объект, получаемый путем вращения сечения вдоль дуги с указанием концевой поверхности)	CorrectProfileAssignment (если тип концевой поверхности — IfcDerivedProfileDef, то она должна быть получена путем трансформации исходного сечения, в противном случае их типы должны совпадать и наследоваться от IfcParametrizedProfileDef)	Генерация концевой поверхности путем трансформации исходного сечения

Заметим, что ряд ограничений в индустриальных моделях данных может быть представлен в виде комбинации однотипных или различных паттернов, связанных логическими операциями, допустимыми в языке EXPRESS (AND, OR, XOR, NOT). К простейшим случаям относятся, например, вышеупомянутые неравенства вида $v \neq const$ (см. раздел 3.2) или комбинированный паттерн существования (см. раздел 3.5).

Логические выражения подобных комбинированных ограничений легко преобразуются к ДНФ. Тогда необходимо и достаточно разрешить паттерны в одном из дизъюнктов. Рассмотрим это на конкретном примере.

```
ENTITY IfcFixedReferenceSweptAreaSolid SUBTYPE OF
(IfcSweptAreaSolid);
```

```
      Directrix : IfcCurve;
      StartParam : OPTIONAL IfcParameterValue;
      EndParam : OPTIONAL IfcParameterValue;
      FixedReference : IfcDirection;
 WHERE
      DirectrixBounded : (EXISTS(StartParam) AND
EXISTS(EndParam)) OR
      (SIZEOF(['IFC4.IFCCONIC',
'IFC4.IFCBOUNDEDCURVE']*TYPEOF(Directrix)) = 1);
END_ENTITY;
```

Объектный тип `IfcFixedReferenceSweptAreaSolid` используется для представления твердых тел, получаемых путем перемещения сечений вдоль директриссы. В нем локальное правило `DirectrixBounded` устанавливает, что либо следует установить параметры начала и конца перемещения по директриссе, либо она должна быть кривой с конечной длиной или коническим сечением. Логическое выражение данного правила может быть переформулировано с использованием разработанной библиотеки следующим образом:

```
IsExist(StartParam) AND IsExist(EndParam) OR
RefinesType(Directrix, ['IFC4.IFCCONIC',
'IFC4.IFCBOUNDEDCURVE'], TRUE);
```

Тогда при установлении ассоциативной связи `Directrix` необходимо и достаточно ограничить множество объектов-кандидатов классами конических сечений или кривых с конечной длиной. При наличии таких кандидатов параметры `StartParam` и `EndParam` могут иметь неопределенные значения. При отсутствии кривых вышеперечисленных типов ассоциативная связь может быть установлена с произвольной кривой, но тогда необходимо сгенерировать значения для `StartParam` и `EndParam`.

5. Заключение

Таким образом, проведен анализ спецификаций индустриально значимого семейства объектно-ориентированных моделей данных (в частности, IFC, CIS/2, общих ресурсов STEP) и выделены паттерны ограничений целостности, используемые в них. Разработана библиотека обобщенных функций на языке EXPRESS для представления каждого из паттернов. Рассмотрена возможность применения данной библиотеки для решения задачи верификации моделей. Для каждого из выделенных паттернов предложен метод разрешения соответствующих ограничений.

Примечательно, что для большинства из паттернов эти методы оказались тривиальными и связанными с ограничением области определения при генерировании значений зависимых атрибутов.

Результаты проведенного исследования демонстрируют важные возможности унификации спецификаций индустриальных моделей данных, которые позволяют улучшить их наглядность, облегчить их дальнейшее сопровождение и развитие и, в целом, ускорить их разработку и упростить анализ с применением автоматизированных средств программной инженерии. Использование паттернов ограничений в спецификациях моделей может служить важной рекомендацией для индустриальных консорциумов и технических комитетов, занимающихся их разработкой и стандартизацией.

Литература

[1]. Model Driven Architecture: The Architecture of Choice for a Changing World. Executive Overview, February 2014. http://www.omg.org/mda/executive_overview.htm

[2]. Unified Modeling Language (UML), V2.4.1, Release Date: August 2011. http://www.omg.org/spec/UML/2.4.1

[3]. Meta Object Facility (MOF) Core, V2.4.2, Release Date: April 2014. http://www.omg.org/spec/MOF/2.4.2

[4]. XML Metadata Interchange (XMI), V2.4.2, Release Date: April 2014. http://www.omg.org/spec/XMI/2.4.2

[5]. Common Warehouse Metamodel (CWM), V1.1, Release Date: March 2003. http://www.omg.org/spec/CWM/1.1

[6]. ISO 10303-1:1994. Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.

[7]. ISO 10303-11:2004. Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.

[8]. ISO 10303-21:2002. Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure.

[9]. ISO 10303-28:2007. Industrial automation systems and integration — Product data representation and exchange — Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas.

[10]. ISO/TS 10303-26:2011. Industrial automation systems and integration — Product data representation and exchange — Part 26: Implementation methods: Binary representation of EXPRESS-driven data.

[11]. ISO 10303-22:1998. Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface.

[12]. ISO 10303-203:1994. Industrial automation systems and integration — Product data representation and exchange — Part 203: Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies.

[13]. ISO 10303-210:2014. Industrial automation systems and integration — Product data representation and exchange — Part 210: Application protocol: Electronic assembly, interconnect and packaging design.

[14]. ISO 10303-212:2001. Industrial automation systems and integration — Product data representation and exchange — Part 212: Application protocol: Electrotechnical design and installation.
[15]. ISO 10303-214:2001. Industrial automation systems and integration — Product data representation and exchange — Part 214: Application protocol: Core data for automotive mechanical design processes.
[16]. ISO 10303-215:2004. Industrial automation systems and integration — Product data representation and exchange — Part 215: Application protocol: Ship arrangement.
[17]. ISO 10303-216:2003. Industrial automation systems and integration — Product data representation and exchange — Part 216: Application protocol: Ship moulded forms.
[18]. ISO 10303-218:2004. Industrial automation systems and integration — Product data representation and exchange — Part 218: Application protocol: Ship structures.
[19]. ISO 10303-233:2012. Industrial automation systems and integration — Product data representation and exchange — Part 233: Application protocol: Systems engineering.
[20]. ISO 10303-236:2006. Industrial automation systems and integration — Product data representation and exchange — Part 236: Application protocol: Furniture catalog and interior design.
[21]. IFC4 Release Summary, March 2013. http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4-release/ifc4-release-summary
[22]. Khemlani L. The CIS/2 Format: Another AEC Interoperability Standard. // AECbytes Newsletter, July 27, 2005. http://www.aecbytes.com/buildingthefuture/2005/CIS2format.html
[23]. ISO 15926-1:2004. Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 1: Overview and fundamental principles.
[24]. CALS-стандарты. // Автоматизация проектирования, № 2, 1997. http://www.osp.ru/ap/1997/02/13031610
[25]. Семенов В.А., Морозов С.В., Тарлапан О.А. Инкрементальная верификация объектно-ориентированных данных на основе спецификации ограничений. // Труды Института системного программирования / под ред. В.П. Иванникова, т. 8, ч. 2, 2004, с. 21–52.
[26]. Семенов В.А., Ерошкин С.Г., Караулов А.А., Энкович И.В. Семантическая реконсиляция прикладных данных на основе моделей. // Труды Института системного программирования / под ред. В.П. Иванникова, т. 13, ч. 2, 2007, с. 141–164.
[27]. ISO 10303-14:2005. Industrial automation systems and integration — Product data representation and exchange — Part 14: Description methods: The EXPRESS-X language reference manual.
[28]. Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT), V1.2, Release Date: February 2015. http://www.omg.org/spec/QVT/1.2
[29]. XSL Transformations (XSLT) Version 1.0. W3C Recommendation, November 1999. http://www.w3.org/TR/xslt
[30]. Nell J. STEP on a page, April 2003. http://www.mel.nist.gov/sc5/soap
[31]. buildingSmart® — International home of openBIM®, October 2014. http://www.buildingsmart.org
[32]. ISO 16739:2013. Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.

[33]. ISO 10303-41:2005. Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support.
[34]. ISO 10303-42:2003. Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation.
[35]. ISO 10303-43:2000. Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resource: Representation structures.
[36]. ISO 10303-45:2008. Industrial automation systems and integration — Product data representation and exchange — Part 45: Integrated generic resource: Material and other engineering properties.
[37]. ISO 10303-46:2011. Industrial automation systems and integration — Product data representation and exchange — Part 46: Integrated generic resource: Visual presentation.
[38]. Staab S., Studer R. (eds.). Handbook on Ontologies, Second Edition. Springer-Verlag, Berlin Heidelberg, 2009.
[39]. Семенов В.А., Морозов С.В., Ильин Д.В. Комбинированный метод верификации масштабных моделей данных. // Труды Института системного программирования / под ред. В.П. Иванникова, т. 26, вып. 2, 2014, с. 197–230. DOI: 10.15514/ISPRAS-2014-26(2)-9
[40]. Tsang E. Foundations of constraint satisfaction. Academic Press Limited, London & San-Diego, 1993.
[41]. Семенов В.А., Сидяка О.В. Теоретические и экспериментальные оценки сложности методов локального распространения в задачах программирования в ограничениях. // Труды Института системного программирования / под ред. В.П. Иванникова, т. 19, 2010, с. 117–134.
[42]. Роджерс Д., Адамс Дж. Математические основы машинной графики. М.: Мир, 2001.

A Constraint Library for Specification of Industrial Data Models

[1, 2]S.V. Morozov <serg@ispras.ru>
[1]D.V. Ilyin <denis.ilyin@ispras.ru>
[1, 3]V.A. Semenov <sem@ispras.ru>
[1, 2]O.A. Tarlapan <oleg@ispras.ru>
[1] Institute for System Programming of the Russian Academy of Sciences, 25, Alexander Solzhenitsyn st., 109004, Moscow, Russia
[2]Lomonosov Moscow State University, 2nd Education Building, Faculty CMC, GSP-1, Leninskie Gory, Moscow, 119991, Russian Federation
[3]Moscow Institute of Physics and Technology, 9 Institutskiy per., Dolgoprudny, Moscow Region, 141700, Russia

Abstract. The paper is addressed to an analysis of object-oriented data models specified at EXPRESS language and widely used in industrial applications. These models play important role for achievement of software interoperability and system integration in accordance with STEP standard family (ISO 10303). The examples of such models are STEP application protocols for machinery construction, automobile industry, shipbuiling, electronics, electrical engineering, systems engineering, furniture production as well as IFC (ISO 16739) model for architecture, engineering and construction, CIS/2 model for manufacturing using constructional steel work, POSC Caesar (ISO 15296) model for oil and gas producing industry. The purpose of the performed analysis is to unify representation of data integrity constraints typically used in the models by means of identification of constraint patterns. The identified patterns are specified at EXPRESS language as an unified constraint library that can be applied both on refactoring of the existing models and on development of new ones. Utilizing the constraint library users can improve clearness of the specifications, to simplify their maintenance and evolution and, on the whole, to accelerate their development. Besides, CASE tools can be effectively applied to analyze the specifications in highly automatic way. Possibility to apply the library for verification of the data models is also discussed in the paper. Rules for resolving the appropriate constraints have been proposed for each pattern. The constraint library can be recommended to industrial consortiums and technical committees that are engaged in development and standardization of the data models. The work is supported by RFBR (grant 13-07-00390).

Keywords: object-oriented modeling, EXPRESS, STEP, IFC, CIS/2, constraint patterns, model verification

DOI: 10.15514/ISPRAS-2015-27(4)-5

For citation: Morozov S.V., Ilyin D.V., Semenov V.A., Tarlapan O.A. A Constraint Library for Specification of Industrial Data Models. *Trudy ISP RAN/Proc. ISP RAS*, vol. 27, issue 4, 2015, pp. 69-110 (in Russian). DOI: 10.15514/ISPRAS-2015-27(4)-5.

References

[1]. Model Driven Architecture: The Architecture of Choice for a Changing World. Executive Overview, February 2014. http://www.omg.org/mda/executive_overview.htm
[2]. Unified Modeling Language (UML), V2.4.1, Release Date: August 2011. http://www.omg.org/spec/UML/2.4.1
[3]. Meta Object Facility (MOF) Core, V2.4.2, Release Date: April 2014. http://www.omg.org/spec/MOF/2.4.2
[4]. XML Metadata Interchange (XMI), V2.4.2, Release Date: April 2014. http://www.omg.org/spec/XMI/2.4.2
[5]. Common Warehouse Metamodel (CWM), V1.1, Release Date: March 2003. http://www.omg.org/spec/CWM/1.1
[6]. ISO 10303-1:1994. Industrial automation systems and integration — Product data representation and exchange — Part 1: Overview and fundamental principles.
[7]. ISO 10303-11:2004. Industrial automation systems and integration — Product data representation and exchange — Part 11: Description methods: The EXPRESS language reference manual.
[8]. ISO 10303-21:2002. Industrial automation systems and integration — Product data representation and exchange — Part 21: Implementation methods: Clear text encoding of the exchange structure.
[9]. ISO 10303-28:2007. Industrial automation systems and integration — Product data representation and exchange — Part 28: Implementation methods: XML representations of EXPRESS schemas and data, using XML schemas.
[10]. ISO/TS 10303-26:2011. Industrial automation systems and integration — Product data representation and exchange — Part 26: Implementation methods: Binary representation of EXPRESS-driven data.
[11]. ISO 10303-22:1998. Industrial automation systems and integration — Product data representation and exchange — Part 22: Implementation methods: Standard data access interface.
[12]. ISO 10303-203:1994. Industrial automation systems and integration — Product data representation and exchange — Part 203: Application protocol: Configuration controlled 3D designs of mechanical parts and assemblies.
[13]. ISO 10303-210:2014. Industrial automation systems and integration — Product data representation and exchange — Part 210: Application protocol: Electronic assembly, interconnect and packaging design.
[14]. ISO 10303-212:2001. Industrial automation systems and integration — Product data representation and exchange — Part 212: Application protocol: Electrotechnical design and installation.
[15]. ISO 10303-214:2001. Industrial automation systems and integration — Product data representation and exchange — Part 214: Application protocol: Core data for automotive mechanical design processes.
[16]. ISO 10303-215:2004. Industrial automation systems and integration — Product data representation and exchange — Part 215: Application protocol: Ship arrangement.
[17]. ISO 10303-216:2003. Industrial automation systems and integration — Product data representation and exchange — Part 216: Application protocol: Ship moulded forms.
[18]. ISO 10303-218:2004. Industrial automation systems and integration — Product data representation and exchange — Part 218: Application protocol: Ship structures.

[19]. ISO 10303-233:2012. Industrial automation systems and integration — Product data representation and exchange — Part 233: Application protocol: Systems engineering.
[20]. ISO 10303-236:2006. Industrial automation systems and integration — Product data representation and exchange — Part 236: Application protocol: Furniture catalog and interior design.
[21]. IFC4 Release Summary, March 2013. http://www.buildingsmart-tech.org/specifications/ifc-releases/ifc4-release/ifc4-release-summary
[22]. Khemlani L. The CIS/2 Format: Another AEC Interoperability Standard. // AECbytes Newsletter, July 27, 2005. http://www.aecbytes.com/buildingthefuture/2005/CIS2format.html
[23]. ISO 15926-1:2004. Industrial automation systems and integration — Integration of life-cycle data for process plants including oil and gas production facilities — Part 1: Overview and fundamental principles.
[24]. CALS-standarty [CALS standards]. // Avtomatizatsiya proektirovaniya [Computer-aided design], no. 2, 1997. http://www.osp.ru/ap/1997/02/13031610 (in Russian).
[25]. Semenov V.A., Morozov S.V., Tarlapan O.A. Inkremental'naya verifikatsiya ob'ektno-orientirovannyh dannyh na osnove spetsifikatsii ogranichenij [Incremental verification of object-oriented data based on specification of constraints]. Trudy ISP RAN [The Proceedings of ISP RAS], vol. 8, no. 2, 2004, pp. 21–52 (in Russian).
[26]. Semenov V.A., Eroshkin S.G., Karaulov A.A., Enkovich I.V. Semanticheskaya rekonsilyatsiya prikladnyh dannyh na osnove modelej [Model-based semantic reconciliation of applied data]. Trudy ISP RAN [The Proceedings of ISP RAS], vol. 13, no. 2, 2007, pp. 141–164 (in Russian).
[27]. ISO 10303-14:2005. Industrial automation systems and integration — Product data representation and exchange — Part 14: Description methods: The EXPRESS-X language reference manual.
[28]. Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT), V1.2, Release Date: February 2015. http://www.omg.org/spec/QVT/1.2
[29]. XSL Transformations (XSLT) Version 1.0. W3C Recommendation, November 1999. http://www.w3.org/TR/xslt
[30]. Nell J. STEP on a page, April 2003. http://www.mel.nist.gov/sc5/soap
[31]. buildingSmart® — International home of openBIM®, October 2014. http://www.buildingsmart.org
[32]. ISO 16739:2013. Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.
[33]. ISO 10303-41:2005. Industrial automation systems and integration — Product data representation and exchange — Part 41: Integrated generic resource: Fundamentals of product description and support.
[34]. ISO 10303-42:2003. Industrial automation systems and integration — Product data representation and exchange — Part 42: Integrated generic resource: Geometric and topological representation.
[35]. ISO 10303-43:2000. Industrial automation systems and integration — Product data representation and exchange — Part 43: Integrated generic resource: Representation structures.
[36]. ISO 10303-45:2008. Industrial automation systems and integration — Product data representation and exchange — Part 45: Integrated generic resource: Material and other engineering properties.

[37]. ISO 10303-46:2011. Industrial automation systems and integration — Product data representation and exchange — Part 46: Integrated generic resource: Visual presentation.
[38]. Staab S., Studer R. (eds.). Handbook on Ontologies, Second Edition. Springer-Verlag, Berlin Heidelberg, 2009.
[39]. Semenov V.A., Morozov S.V., Ilyin D.V. Kombinirovannyj metod verifikatsii masshtabnyh modelej dannyh [A combined method for verification of large-scale data models]. Trudy ISP RAN [The Proceedings of ISP RAS], vol. 26, no. 2, 2014, pp. 197–230 (in Russian). DOI: 10.15514/ISPRAS-2014-26(2)-9.
[40]. Tsang E. Foundations of constraint satisfaction. Academic Press Limited, London & San-Diego, 1993.
[41]. Semenov V.A., Sidyaka O.V. Teoreticheskie i eksperimental'nye otsenki slozhnosti metodov lokal'nogo rasprostraneniya v zadachah programmirovaniya v ogranicheniyah [Theoretical and practical complexity estimates for local propagation methods in constraint-based programming applications]. Trudy ISP RAN [The Proceedings of ISP RAS], vol. 19, 2010, pp. 117–134 (in Russian).
[42]. Rogers D.F., Adams J.A. Mathematical Elements of Computer Graphics, Second Edition. McGraw Hill, 1990.